# Trabalho 08

## Nome da Atividade: Aproximação Funcional

### Nome e Matrícula: Lucas Lima do Nascimento - 12111ECP024

Treinamento:

```
%treinamento irá parar pelo número de ciclo ou pelo errototal alcançado.
while (ciclo < numciclo) && (errototal > errototaladmissivel)
    ciclo=ciclo+1;
    errototal=0;


    for padroes=1:10
       % -----------fase feedforward --------------
    % inserção de cada padrao de treinamento na entrada da rede neural
    % e cálculo das saídas z dos neurônios escondidos
         for j=1:neuroniosescondidos
             zin(j)= x(padroes)*v(j)+bv(j);
             z(j) = (2/(1+exp(-zin(j))))-1;
         end
        %%Cálculo da  saída  da rede
         yin=z*w+bw;
         y=(2/(1+exp(-yin)))-1;

    % -------------fase da Retropropagação do erro----------
    % da saida para a camada escondida
    %cálculos do deltinhaw do neurônio de saída
      deltinhaw = (t(padroes) - y)*0.5*(1+y)*(1-y);
   % cálculo dos deltaw para atualização dos pesos do neurônio de
   % saida
      for j=1:neuroniosescondidos
        deltaw= alfa*deltinhaw*z(j);
      end
```

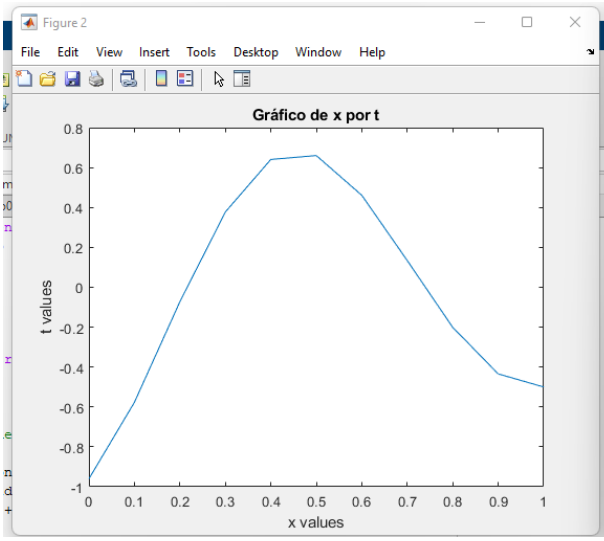
```
  %cálculo das atualizações dos bias dos neurônios de saida
     deltabw=alfa*deltinhaw;

 % cáculo dos deltinhav da camada escondida para as atualizações dos pesos
 % dos neurônios escondidos
    for j=1:neuroniosescondidos
      deltinhav(j)=deltinhaw*w(j)*0.5*(1+z(j))*(1-z(j));
    end

%cálculo dos deltav para atualização dos pesos dos neurônios escondidos
    for i=1:neuroniosentrada
      for j=1:neuroniosescondidos
       deltav(i,j)= alfa*deltinhav(j)*x(padroes,i);
      end
    end

% cálculo dos deltabv, bias dos neurônios escondidos
    for i=1:neuroniosescondidos
      deltabv(i)= alfa*deltinhav(i);
    end

%-------atualização dos pesos----------
% dos neurônios da camada de saída
    w=w+deltaw
    bw=bw+deltabw;
% dos neurônios da camada escondida
   for i=1:neuroniosentrada
```

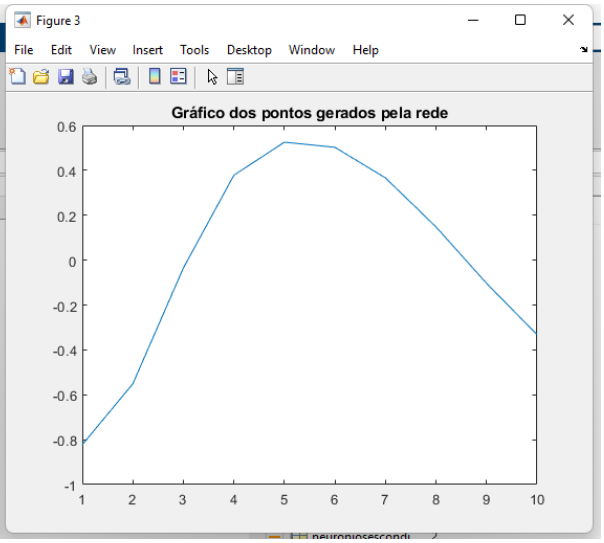
```
     for j=1:neuroniosescondidos
       v(i,j)= v(i,j)+deltav(i,j);
     end
   end
   for i=1:neuroniosescondidos
     bv(i)=bv(i)+deltabv(i);
   end

   % cálculo do erro total
   % é a soma de todos os erros  produzidos pelos neurônios de saida
   % para todos os padrões de treinamento
   errototal=errototal+0.5*((t(padroes)-y)^2);
   end % for padrões de entrada
    %plot(ciclo,errototal,'r.');
    erroquadraticototal(ciclo)=errototal;
end % while
```

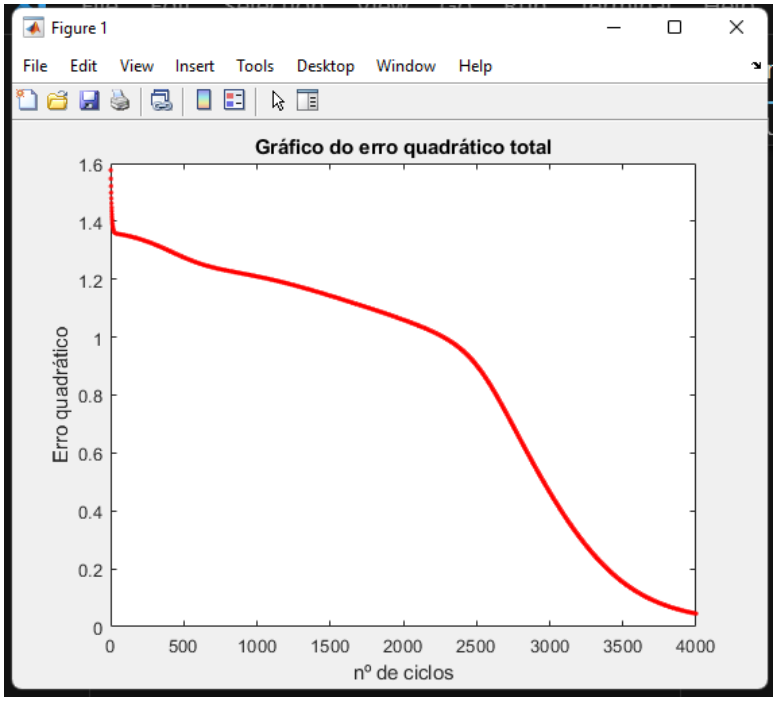
Teste e Resultado:

```
Teste da rede treinada
t: -0.960200   y: -0.471775
t: -0.577000   y: -0.306205
t: -0.072900   y: -0.145926
t: 0.377100   y: -0.014997
t: 0.640500   y: 0.080067
t: 0.660000   y: 0.144740
t: 0.460900   y: 0.187548
t: 0.133600   y: 0.215739
t: -0.201300   y: 0.234427
t: -0.434400   y: 0.246966
```

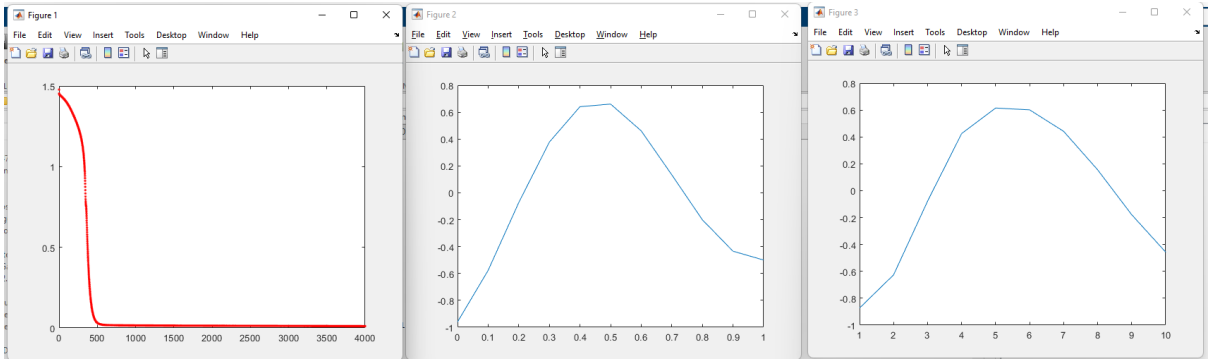
```
%--------teste da rede com os padrões de treinamento ---------
    for padroes=1:10
        for j=1:neuroniosescondidos
         zin(j)= x(padroes,:)* v(:,j)+bv(j);
         z(j) = (2/(1+exp(-zin(j))))-1;
        end
        %Cálculo da saída  da rede
         yin=z*w+bw;
         y=(2/(1+exp(-yin)))-1;
         final_values(padroes) = y;
         fprintf("t: %f   y: %f\n",t(padroes),y);
    end
```



Alterando valores de alfa para 0.5 e o número de neurônios escondidos para 10, obtive uma queda acentuada no erro:




# Meu código completo:

Em anexo ao envio